Truck Platooning: week 6

Changyao Zhou, Jiaxin Pan

February 1, 2022

1 Introduction of the project

The goal of this project is to chase a chosen target vehicle by predicting the steering angle and throttle input of the ego-vehicle, given the velocity of both vehicles and images taken from cameras.

The control values (steering angle and throttle) computed by Model Predictive Control (MPC) are used as ground-truth. MPC is an advanced method of process control that is used to control a process while satisfying a set of constraints[WIK]. It takes a prediction horizon instead of a single time step into account, and aims to get an optimal control result minimizing the objective function within the prediction horizon. For learning the MPC controller, we take [MPC] as reference.

The project is designed to be realized in the following steps:

1). Control the ego-vehicle with Model Predictive Control (MPC) in the Carla simulator

2). Given x, y-location, yaw angle and velocity of both vehicles, train a fully connected neural network to control ego-vehicle by using the MPC output as ground-truth.

2). Given object velocity, ego-velocity and RGB image, train a neural network to control the ego-vehicle by using the MPC output as ground-truth

3). Further improvement by changing the input of the neural network

2 Overview of this week

Last week we have tried to train the FCNN with delta state of two vehicles as input, as well as CNN with images input. Unfortunately, we couldn't get result as stable as before. The ego vehicle can roughly follow the target vehicle, but not as perfectly as putting the raw states as input.

This week, we have collected more data from more trajectories in the same map, in order to make the model better generalize with a much larger dataset. Also, we tried different possible architectures of the FCNN, which is fed with delta state as input, to see if the delta state input is enough for prediction of two control values. In addition, we also trained a smaller architecture of CNN.

All results of models mentioned above were tested with the 'online' mode in CARLA, where the target vehicle is set in 'autopilot' mode. All the videos are stored under `https://syncandshare.lrz.de/getlink/fiXzt51MtSVwwKGMGeHnvN54/week6`.

3 Data Collection

In order to make the model to better cope with various road conditions, we have collected trajectories with different situations, such as going straight, slowly turning left and right, sharply turning left and right, ring road in Town04 map of CARLA. The following figure shows whole map of town04. And the blue lines are trajectories we collected, including straight road, slow turn, sharp turn and ring road. We can see that the town04 map basically consist of surrounding highways and internal trails. We tried to include every possible road conditions in our dataset. If our model could ideally generalize well, then the ego-vehicle can follow the target vehicle, no matter where the start point is on the map.

The dataset is collected online with CARLA simulator and it composes of on-trajectory part and off-trajectory part. And for each sample point, the states of both vehicles, i.e. the x- and y- position, yaw angle, velocity as well as the ground truth throttle and steering angle are collected.

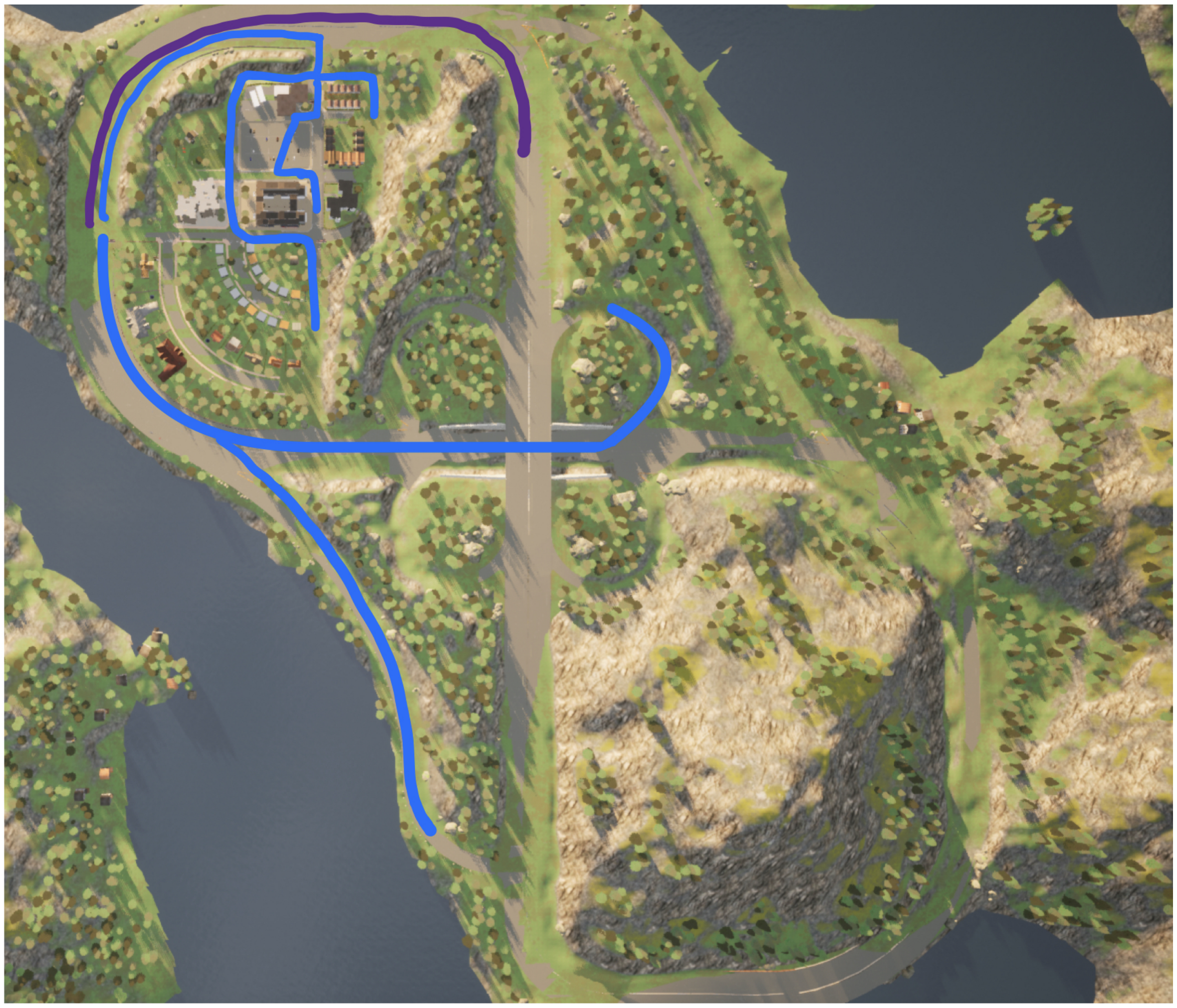

Figure 1: Town04 map of CARLA and collected trajectories

We tested the CNN Model we trained last week with 2 ego vehicles and the performance of the second ego vehicle is not that good as the first ego-vehicle. The reason might be that in the image it captured both the target vehicle and the first ego vehicle are visible, which is different from the images collected by the first ego vehicle. At the same time, the image taken from the camera attached on the ego-vehicle is also saved. This time we put the camera at a lower height, so that there is no car hood part in the images and even for the second ego vehicle, only the vehicle in front of it can be seen. The Figure 2 shows the image taken from camera.

Until now, we've collected 63088 samples around town04, including 4527 on-trajectory samples and 58561 off-trajectory samples.

4 Training of the FCNN Model

After collecting the dataset, we now started to train the FCNN with the delta state and the velocities of both vehicles as input. We tried different layers and architectures of the FCNN and the minimal train loss could reach 4e-4 and validation loss could reach 3e-2. However, when testing the trained model in CARLA, the performance seems not so good. The ego-vehicle controlled by predicted throttle and steering angle could follow the target vehicle in the straight line, but the routes of both vehicles will be very different when turning. We also compare the ground truth and the prediction result. It seems that the throttles are very close to the ground truth, but the steering angles always have an obvious difference.

The input Δyaw angle seems to have a big affect on the prediction of steering angle. However, the

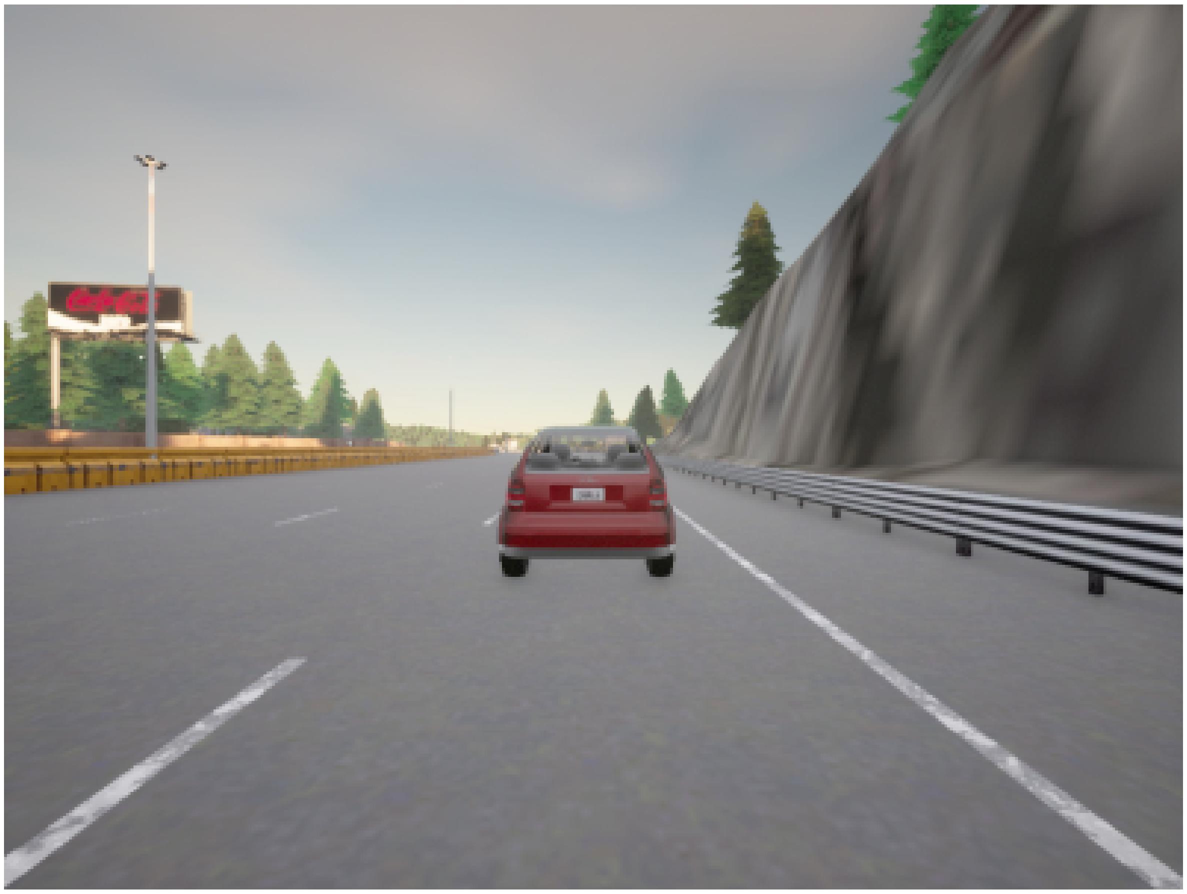

Figure 2: Image from RGB Camera

difference between two yaw angles of both vehicles is very small at most time. This is probably one of the reasons of such result.

After many attempts we still can't get a stable result with delta states. Then we turn to use two yaw angles instead of Δyaw angle as input, which means the current inputs are Δx, Δy, two yaw angles, and two velocities. This time we could get a good performance after training. From the videos under https://syncandshare.lrz.de/getlink/fi3kSoeSmgLHikViCEitEgbt/delta_2yaw_seen_1.mp4 and https://syncandshare.lrz.de/getlink/fi74dRT8nAhQCqXxEqF8AoAb/delta_2yaw_seen_2.mp4 we can see the ego-vehicle could follow the target vehicle stably. And we also tried to test car following starting from a starting point outside the dataset (the purple line shown in figure 2) and the performance is also good. https://syncandshare.lrz.de/getlink/fiSr4LUVNwkZcnY7bXSiwZzK/delta_2yaw_unseen.mp4 That means the model can generalize well over the dataset.

In addition, we tried to do two-car platooning with the trained model mentioned above. The result is saved under https://syncandshare.lrz.de/getlink/fiSBvWU2cohBViPaokbZLfAX/delta_2yaw_2fol.mp4. We can see that due to the vibration of the target vehicle, the following two vehicles behind shake more violently when following. But the car following performance is generally good.

5 Training of CNN Model

Instead of the pretrained alexnet, this week we used a much smaller architecture shown in the table 1.

After training this CNN model, the training loss and validation loss could reach 4e-2 and 6e-2 respectively. But it still can't implement a stable car following in CARLA.

Table 1: CNN Architecture

Layer Number	Layer Type	Layer Input	Layer Output
1	Convolution	3x200x150	30x100x75
2	MaxPooling	30x100x75	30x50x37
3	ReLU activation	30x50x37	30x50x37
4	Convolution	30x50x37	60x25x18
5	MaxPooling	60x25x18	60x12x9
6	ReLU activation	60x12x9	60x12x9
7	Convolution	60x12x9	60x6x4
8	MaxPooling	60x6x4	60x3x2
9	Flatten	62x3x2	372
10	Linear Layer	372	32
11	ReLU activation	32	32
12	Linear Layer	32	2

6 Conclusion and future work

In this week, we have collected more data around the whole map to help the model deal with various road condition. However, we didn't get any better result with FCNN, which takes delta state and velocities of both vehicles as input, and CNN, which takes image and velocities of both vehicles as input. Thus, we wonder if the delta state could provide sufficient information for control value prediction.

The next step is to try more different architectures of CNN to improve its performance.

References

[MPC] Model Predictive Control. https://github.com/WuStangDan/mpc-course-assignments.

[WIK] Model Predictive Control. https://en.wikipedia.org/wiki/Model_predictive_control.